THE BELL SYSTEM
TECHNICAL JOURNAL

Volume 47 April 1968 Number 4

Some Theorems and Procedures for Loop-Free Routing in Directed Communication Networks

By R. MAGNANI

(Manuscript received November 15, 1967)

This paper examines two general methods for specifying directed routing patterns in communication networks. Hierarchical routing, as currently used in the toll network, is such a directed routing pattern. However, it is only one member of a large set of possible routing strategies that can be realized by storing, in each office, a list of outgoing trunk groups and an order-of-choice for these groups for each received call address. The general class of routing strategies is defined by this method of realization, subject to the constraint that routing patterns be loop-free. The paper discusses procedures for generating loop-free patterns, for detecting whether or not a given pattern is loop-free, and for specifying "good" patterns from the large number which are realizable.

I. INTRODUCTION

The fundamental problem which besets people concerned with the design of communication networks is how to provide a network which is, at once:

(*i*) Of sufficient routing capability to allow any two users to be connected with a high probability of success.

(*ii*) Economical in its use of transmission facilities and switching centers.

(*iii*) Capable of surviving extensive natural or man-made damage.
(*iv*) Adaptable to changing traffic patterns and overload situations.
(*v*) Capable of being engineered and implemented in small sections, over a period of years, by many different people.

This is a problem of such complexity that, with the present state of the theory, it must be attacked piecemeal.

This paper deals with a small but important section of the problem. It considers some of the topological properties of communication networks and examines a class of alternate routing strategies from a general point of view. Our purpose is to state rules which will allow the orderly production of routing patterns, for arbitrary networks, by a computer. We approach the problem in three stages:

(*i*) Several simple rules are stated for producing "loop-free" routing patterns.
(*ii*) The rules are "generalized" to allow the proof of some theorems about the extent of their application.
(*iii*) A more limited but practical statement of the rules is presented followed by several heuristic procedures, based on these rules, which can be used to specify "good" routing patterns from the large number which can be generated.

II. BACKGROUND

What is fundamental to the routing process as it is practiced today in the telephone network? Certainly one of the answers is that each office stores a list of outgoing trunks and an order of choice for those trunks for each possible call address that can be received. We can think of the aggregate of these lists as constituting a "routing map" which has been distributed among many offices. The "map" which is currently stored in the telephone network realizes the hierarchical routing plan.

But suppose we wish to examine strategies which do not require a "hierarchy" of offices? Is there a way to realize a general class of routing maps which will allow offices to be of *equal* rank and which can be implemented in the same fashion as the current hierarchical plan? The answer is that such a class of routing maps does exist and that, indeed, the present hierarchical map is simply one member of the set. To see this, consider the simple network of four offices shown in Fig. 1. This may be an entire network or some subset of a much larger network with the connecting trunk groups omitted. For the

present, we will assume the trunk groups shown are all two-way (that is, contain trunks that can be seized from either end) and that routing between the offices is subject to the following constraints:

(*i*) No routing control information is passed between offices.

(*ii*) Offices do not check for shuttle.*

The resulting network is a fair approximation to a subset of the present day commercial network.† Routing between offices is accomplished as follows :

(*i*) Each office is assigned a unique address such as NNX or NPA-NNX.

(*ii*) Upon receiving a call request, an office checks to see if it is the destination office. If it is, the call is counted as a success (although in practice the called subscriber's line must still be checked). If not, the office consults a routing table and, on the basis of the received NNX, selects an outgoing trunk group.

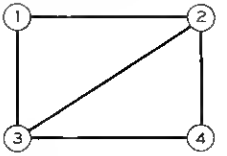

Fig. 1 — A four-office network.

(*iii*) If a trunk in the group is available, it is seized and the called address is passed over it to the next office. At this point we return to step (*ii*).

(*iv*) If no trunks are available, the office again consults its routing table, to find an alternate trunk group, and returns to step *iii*. The process continues until all alternate trunk groups have been tried.

(*v*) If no trunks are available in any of the alternate trunk groups, the call is blocked, and the caller is so notified.

For a particular network and destination office, this process may be conveniently summarized on the graph which represents the network; for routing to a particular office, we assign each trunk group in the network a direction and an order of choice out of the office in which it originates. For example, if office 4 were the destination office in the

* Shuttle refers to routing a call out over the same trunk group on which the call arrived.

† With one-way trunk groups omitted.

network of Fig. 1, we might summarize routing to this office by the use of Fig. 2.

Here we are to understand that calls must route in the directions shown and that trunk groups are to be selected in the given order (beginning with 1). The routes between office 1 and office 4 can easily be seen to be: 1–2–4, 1–3–4, and 1–3–2–4, where the numbers represent *office* numbers and the routes are listed in the order they will occur. Similarly, offices 2 and 3 can be seen to have routes 3–4, 3–2–4, and 2–4.

Fig. 2 represents routing from *all offices* to office 4 and will be said to be a *directed routing pattern* to office 4 on the network of Fig. 1. When such a routing strategy is followed, the way in which a call routes from a particular office is independent of the past history of the call; this is characteristic of routing in the present DDD network. To

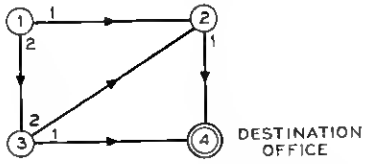

Fig. 2 — A directed routing pattern to office 4.

completely specify routing in the network of Fig. 1, four directed routing patterns are necessary, one with each office as destination. The direction and order of choice assigned to the trunk groups will vary from pattern to pattern depending on the destination office. To see that the hierarchical pattern in use today is of this type, we need simply draw the pattern. (See Fig. 3.)

III. ROUTING PATTERNS

Let us examine some simple rules for constructing such patterns and adopt the standard graph theory terminology: "branch" or "link" for trunk group, and "node" for office.

Because we assume offices do not check for shuttle or looping, we will require the patterns we generate to be loop-free.* A "loop" for our purposes is defined as: a set of branches and nodes (not containing the destination node) constitutes a *loop* if we can select any node

*It has been suggested by J. H. Weber that a small probability of looping may be acceptable if looping can be detected (see Ref. 1).

in the set and, by following the directions of the branches, traverse each branch once to form a path which returns to the selected node (that is, loops must be "directed"). It is not clear in the case of a large network just how one goes about obtaining a loop-free directed routing pattern, particularly if the network is nonplanar. To demonstrate that a systematic procedure is required, we invite you to try to draw a loop-free pattern on the network of Fig. 4a.

A closely related problem is illustrated by Fig. 4b in which we are given a routing pattern and asked to determine whether or not it contains a loop. In this case, the single loop that the network does contain may or may not be obvious to you; however, if the network were much larger, say 40 nodes, a systematic procedure again would be required.

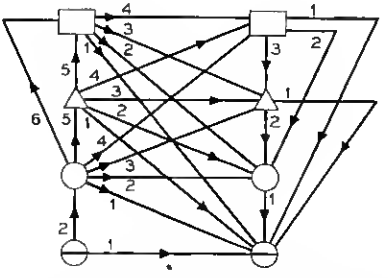

Fig. 3 — Hierarchical directed routing pattern.

Consider the following two rules for generating directed routing patterns in an arbitrary network.

Rule 1—Select any node in the network (usually the destination node) and label all its branches incoming.

Rule 2—Now select a node which has at least one outgoing branch and label all its remaining free branches incoming.* Repeat this rule until all branches in the network have been assigned a direction.

Fig. 5 illustrates the process for a simple 6-node network. The numbers in the node circles represent the order in which rules 1 and 2 are applied, beginning with the heavily circled node that is the destination node for this pattern. Notice that the process is finished in four steps, leaving the two blank nodes with only outgoing branches. We will call nodes of this type (the blank nodes) *originating* nodes, although it is assumed that *calls routing to the destination node can originate at any*

* Free branches are those which have not been given a direction.

node. Nodes with both incoming and outgoing branches will be termed *tandem nodes* (for example, 2, 3, 4). The remaining node, node 1, will be called a *terminating node* and, in this case, it is the destination node for the pattern. Indeed, if the rules remain as stated, there can be only *one* terminating node in any pattern, the destination node. Unless stations are being considered which are multiple-homed, this

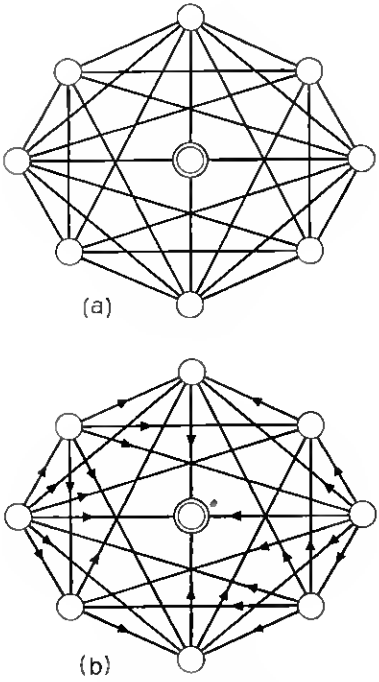

Fig. 4 — A sample pattern.

is not a limitation.* However, to deal with a multiple-homed situation and to allow the proofs of some general theorems, we will remove this restriction by restating rule 1:

Rule 1' Select any free node in the network and label all its branches incoming.† This rule may be repeated an arbitrary number of times, each application creating a terminating node.

*A station which can be reached from more than one office is said to be multiple-homed. A dual-homed station, for example, can be reached from either of two offices—in this case, we would want both offices (if not connected) to be terminating nodes in the routing pattern.

† A free node is one with no directed branches. Each application of this rule, of course, creates a "trap" for traffic.

Rule 2' Same as 2.

This revised set of rules will be referred to as *backward production.*

Clearly an analogous process exists in the forward direction and will be called *forward production.*

Rule 1" Select any free node in the network and label all its branches outgoing. This rule may be repeated arbitrarily, each application creating an originating node.

Rule 2" Now select a node which has at least one incoming branch and label all its remaining free branches outgoing. Repeat rule 2 until all branches in the network have been assigned a direction.

We show later that backward production is the more useful process for generating telephone network routing patterns.

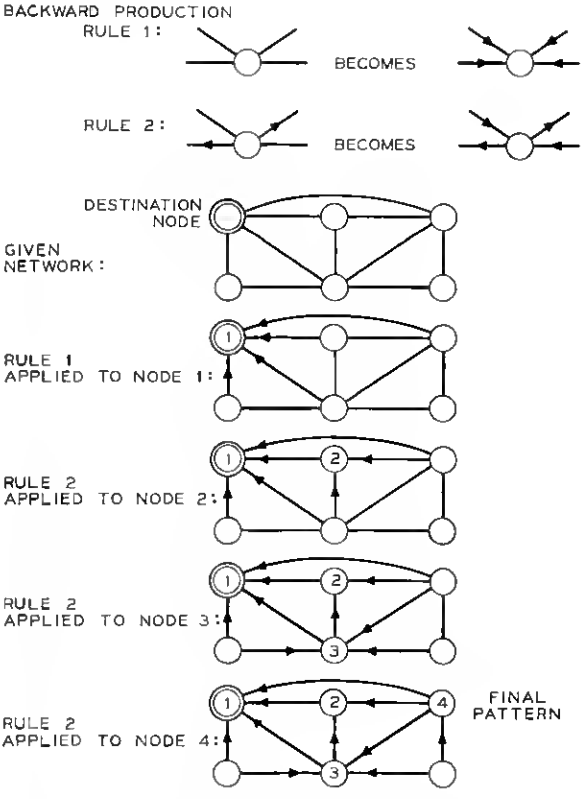

Fig. 5 — Example of use of backward production.

IV. GENERAL THEOREMS

4.1 *Proof of Lemma and Theorems*

We now prove the following lemma and theorems:

*Lemma 1: Any loop-free pattern must contain at least one originating node and at least one terminating node.**

Proof: This can be proven by exhausting the possibilities. Clearly it is not possible to have a network consisting only of originating nodes or only of terminating nodes. The remaining possibilities are:

(*i*) *Only Tandem Nodes*—If A is a tandem node, it must have an outgoing branch to some other node, B. Similarly, B must have an outgoing branch to some node other than A (or we would have a loop), say C. This argument proceeds until all nodes in the network have been considered. The last node to be considered *must* connect to some previous node since it, too, is a tandem node. Such a connection would create a loop.

(*ii*) *Originating Nodes and Tandem Nodes*—If A is a tandem node, its outgoing branches must connect to another tandem node, since no branches can terminate on an originating node. Therefore, the argument presented in *i* can be used here.

(*iii*) *Terminating Nodes and Tandem Nodes*—If A is a tandem node, its incoming branches must originate at another tandem node, say B, since no branches originate at terminating nodes. Since B is also a tandem node, it must have incoming branches from some node other than A (or we would have a loop), say C. Again, this argument proceeds until all nodes in the network have been considered. The last node considered must have an incoming branch from some previously considered tandem node, thus creating a loop.

The remaining two possibilities (terminating the originating nodes only, and all three node types), each contain at least one terminating and one originating node.

Theorem 1: Routing patterns generated by backward production are loop-free.

Proof: Rule 1′ tells us we may create terminating nodes arbitrarily (we must create at least one) in sequential fashion. Since candidates for terminating nodes must have all branches free and these branches

*As this paper was being prepared, the author learned of work by S. L. Hakimi in which Lemma 1 and Theorems 1 and 2 are proved in a more formal fashion. (See Ref. 2.)

are all made incoming, it is not possible to loop through a terminating node, nor can there be branches between terminating nodes. Let A and B be terminating nodes and let C be the first nonterminating node, with a branch to A (and possibly to B), to which we apply rule 2′. Since it is not possible to loop through nodes A and B, we may disregard the branches to these nodes as far as loops are concerned. Then the only branches which can be members of a loop are the remaining (all-free) branches on C. But rule 2′ tells us to make all these branches incoming. Therefore, the only way to loop through node C is to loop through node A (or B). Since it is not possible to loop through A or B, it is not possible to loop through C. Clearly the same argument applies at each stage of the process; the only way to loop through the present node is to loop through some previously considered node, which is not possible. The process ends when all branches have been given a direction. At this point, the originating nodes will be seen to have been created by applying rule 2′ to all the nodes to which they connect. Since it is not possible to loop through originating nodes, the pattern must, indeed, be loop-free.

By a completely analogous proof, it may be shown that the routing patterns generated by forward production are also loop-free.

Now we have two procedures for generating loop-free patterns. The question is: What sort of patterns do they generate? We prove:

Theorem 2: All loop-free routing patterns can be generated by backward production.

Proof: This is proven by induction on Lemma 1. Let N_0 be any arbitrary loop-free routing pattern. Then, by Lemma 1, it must contain at least one terminating node. For generality, assume it contains two, A and B. We will make these nodes evident, leaving a reduced network, N_1. See Fig. 6.

In a blank network (that is, a copy of N_0 without branch assign-

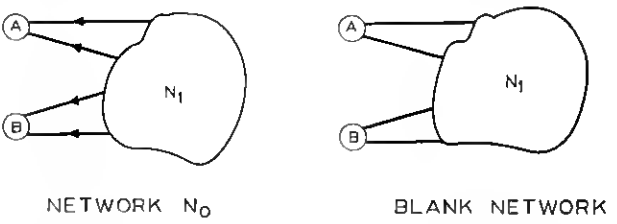

Fig. 6 — Construction of a duplicate of N_0 from a blank network—first stage.

ments), the terminating nodes A and B may be generated by the application of rule 1′.

If we were to remove these nodes and their branches from N_0, we would have left the network N_1, which must also be loop free. (If N_1 is not loop-free, then neither is N_0.) Since N_1 did not contain terminating nodes (all terminating nodes in N_0 were made evident), we must create terminating nodes in the process of discarding nodes A and B and their branches. That is, in the set of nodes to which A and B connect, there must be at least one node which becomes a terminating node if its branches to A and B are removed. In addition, if there is more than one such node, the nodes cannot be connected to each other. (Any such connection would make one of the nodes a nonterminating node.) We will call these terminating nodes, generated by discarding previous nodes and branches, *pseudoterminating*. Let us assume there are two such nodes, C and N, in the network N_1 and place them in evidence. See Fig. 7. In the blank network, the outgoing branches from C and D (and all other nodes) to A and B, were generated when we applied rule 1′ to nodes A and B. If we were now to apply rule 2′ to nodes C and D, in any order, we would generate the nodes C and D in the blank network exactly as they appear in the network N_0.

If we now remove nodes C and D and their branches from N_1, we are left with network N_2, which must also be loop-free. N_2 had contained only tandem and originating nodes (all pseudoterminating nodes in N_1 were made evident). With the removal of branches to C and N, we must therefore create at least one pseudoterminating node in N_2. Let there be two such nodes, E and F, and make them evident. See Fig. 8. In the blank network, the application of rule 1′ to nodes A and B, and of rule 2′ to nodes C and N, assigned all the outgoing branches from nodes E and F (and all other nodes) to nodes A, B, C, and N. Now the application of rule 2′ to nodes E and F in the blank network will properly assign all the incoming branches to nodes E and F, and these nodes will appear as they do in N_0.

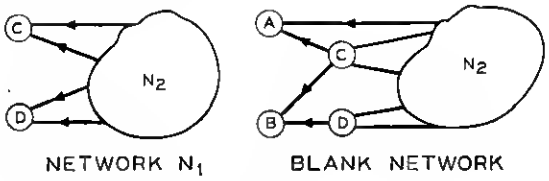

Fig. 7 — Second stage.

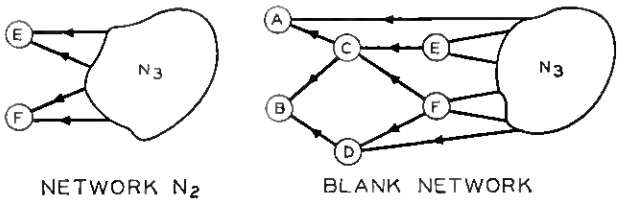

Fig. 8 — Third stage.

This argument may be applied to smaller and smaller networks, each time generating the proper nodes and branch assignments in the blank network. Since the network N_0 is assumed to be finite, the process terminates in K steps with some network, N_K, which contains all the originating nodes in N_0, now isolated (all branches will have been discarded). At this point in our assignments in the blank network, we will find that all branches have been assigned and that we have generated network N_0 by applying rules 1′ and 2′.

Again, an analogous proof will show that all loop-free patterns may be generated by forward production.

It is possible to decide whether or not a given network and routing pattern contains a loop by a procedure which is a variant of that given in theorem 2. (This procedure relies on lemma 1 for its justification.)

Identify all originating and terminating nodes and remove them and all their branches from the graph. Now repeat this step until either:

(*i*) Only branchless nodes remain, or

(*ii*) No originating or terminating nodes can be found. If the network can be reduced to branchless nodes, it is loop free. If not, at the point where no further reduction is possible, the remaining network contains at least one loop.

4.2 *Conclusions from the Theorems*

We can now draw the following conclusions:

(*i*) The class of routing patterns defined by rules 1′ and 2′, or by rules 1″ and 2″, is equal to the class of all loop-free routing patterns.

(*ii*) Therefore, any pattern that can be generated by forward production can also be generated by backward production.

(*iii*) If a pattern is loop free, there is at least one sequence of nodes, to which we can apply backward production, that will generate the

pattern. (A similar statement holds for forward production.) The sequence that will generate the pattern is discoverable by applying the procedure given in theorem 2.

(*iv*) If N_i represents the number of terminating or pseudoterminating nodes made evident in the i^{th} step, the number of ways to generate the pattern N_0, by backward production, is:

$$\text{No. of Ways} = \prod_{i=1}^{K} (N_i\,!)$$

In general, the number of ways to produce a pattern using forward production is unequal to the number of ways using backward production.

V. SYMMETRIC NETWORKS

5.1 *Additional Conclusions*

The theorems and conclusions of Section IV apply equally well to symmetric networks (that is, fully interconnected). In addition, it can be shown that, for symmetric networks, the following is true:

(*i*) There is exactly one originating node and exactly one terminating node in any pattern.

(*ii*) There is only one way to produce a *given* pattern using backward production. (Similarly for forward production.)

(*iii*) If we choose a terminating node and apply backward (or forward) production, we can generate $(N-1)$ distinct loop-free routing patterns to the terminal node. These patterns will all be isomorphic (that is, the same with a relabeling of the nodes). Indeed, there is really only one "abstract" loop-free pattern in a symmetric network, no matter which node is the destination node or what order of choice is applied, since all patterns can be shown to be isomorphic.

5.2 *Routing in Symmetric Networks*

We would now like to examine routing in symmetric networks both as useful in itself and for a bound on routing in incomplete networks. We begin by deriving an expression for the number of K-link routes in a symmetric routing pattern from a given node to the destination node.

Assume we are given an N node symmetric network with the first node as the destination. Also assume forward production is applied to the network, using rule 1 on node N and rule 2 on the nodes $N-1$, $N-2$, $\cdots$ 3, and 2, *in that order*. (Equivalently, we could use backward

production, applying rule 1 to node 1 and rule 2 to nodes 2, 3, 4, $\cdots$ $N - 2$, and $N - 1$, in that order.) Then we may show the following:

Theorem 3: In a symmetric network of N nodes, the number of K-link routes between the Q^{th} node ($2 \leqq Q \leqq N$) and the destination node (node 1) is exactly given by the binomial coefficient $C\binom{Q-2}{K-1}$.

Proof: Consider node N. It will have one branch to the destination node, giving a single one-link path. We may write this as $C\binom{N-2}{0}$. Now, any 2-link path must pass through an intermediate node, of which there are $N - 2$. If we can show that each of the $C\binom{N-2}{1}$ selections of an intermediate node generates exactly one 2-link path, the number of 2-link paths from node N to the terminal node will be $C\binom{N-2}{1}$. Let A be one of the possible intermediate nodes. Since we are using forward production to generate the routing pattern, each successive node considered must have no branches directed toward previously considered nodes and must have exactly one branch directed to each of the nodes not yet considered. Since node A is considered sometime after node N, there must exist a branch from N to A. But A is considered before the destination node (which is last in the process); therefore, there must exist a branch *from A to* the destination node. Thus, there is exactly one 2-link route from node N, through node A, to node 1. This argument is valid for any of the $C\binom{N-2}{1}$ choices for A. Therefore, there are exactly $C\binom{N-2}{1}$ 2-link routes from node N to node 1.

This argument may be generalized for K-link routes. Each K-link route requires $K - 1$ intermediate nodes between node N and node 1. There are $C\binom{N-2}{K-1}$ ways to choose a distinct set of $K - 1$ intermediate nodes. If we let A_1, A_2, A_3, $\cdots$ A_{K-1} be a particular choice of the $K - 1$ nodes, then there must exist exactly one ordering of these nodes which represents the sequence in which rule 2″ was applied. If the ordering is A_1, A_2, A_3, $\cdots$ A_{K-1}, node A_1 will have a branch to A_2, which will have a branch to A_3, and so on. Since A_1 is always considered after node N, and node A_{K-1} is always considered before node 1, there will be exactly one K-link path for this choice of $K - 1$

nodes. Since the choice was arbitrary, there are $C\binom{N-2}{K-1}$ K-link routes from node N to node 1.

Now consider node $N-1$. All branches from node N were made outgoing and therefore are of no use to later nodes for the purpose of routing. If we remove node N and its branches from the graph, we are left with an $N-1$ node symmetric network. In this reduced graph, we may consider node $N-1$ as the originating node and node 1 as the destination node. The sequence in which rule 2″ was applied in this graph is the same sequence used in the larger graph. Hence, the argument presented above applies to this network with $N-1$ nodes replacing N nodes. The number of K-link routes is therefore $C\begin{bmatrix}(N-1)-2\\K-1\end{bmatrix}$. We may proceed to remove node $N-1$ and its branches from the graph, and so on, generating successively smaller symmetric networks and applying the same arguments at each stage. In general, from node Q, the number of K-link routes to the destination is $C\binom{Q-2}{K-1}$, $(2 \leqq Q \leqq N$ and $1 \leqq K \leqq Q-1)$. As a corollary, it can be shown that the *total* number of K-link routes from all nodes in the graph to the destination node is given by $C\binom{N-1}{K}$. That is:

$$C\binom{N-1}{K} = \sum_{Q=2}^{N} C\binom{Q-2}{K-1}$$

where

$$C\binom{Q-2}{K-1} \text{ is zero for } K \geqq Q.$$

It is possible to summarize routing in symmetric networks by using Table I. As an example of the information obtainable from the table, consider a 6-node symmetric network to which we have applied backward production in the order: (rule 1) 1, (rule 2) 2, 3, 4, 5, 6. (See Fig. 9.)
This network will have:

From node 6: one 1-link, four 2-link, six 3-link, four 4-link, and one 5-link route to node 1 (the destination).
From node 5: one 1-link, three 2-link, three 3-link, and one 4-link route to node 1.
And so on, reading node I routes from line I. Reading from $N=6$,

TABLE I—SUMMARY OF ROUTING IN SYMMETRIC NETWORKS

Number of K-Link Routes From Node I to Node 1 (Terminal Node)	Number of Routes											
I	1 Link	2 Link	3 Link	4 Link	5 Link	6 Link	7 Link	8 Link	9 Link	10 Link	11 Link	N
2	1											
3	1	1										2
4	1	2	1									3
5	1	3	3	1								4
6	1	4	6	4	1							5
7	1	5	10	10	5	1						6
8	1	6	15	20	15	6	1					7
9	1	7	21	35	35	21	7	1				8
10	1	8	28	56	70	56	28	8	1			9
11	1	9	36	84	126	126	84	36	9	1		10
12	1	10	45	120	210	252	210	120	45	10	1	11
		1 Link	2 Link	3 Link	4 Link	5 Link	6 Link	7 Link	8 Link	9 Link	10 Link	N
		Total Number of Routes										Total Number of K-Link Routes in a Network of N Nodes

there will be a total of five 1-link routes from all nodes to node 1, a total of ten 2-link routes, and so on.

These numbers represent the maximum numbers of K-link routes in an arbitrary (not necessarily symmetric) 6-node network. This follows from the fact that we may generate the arbitrary network by removing branches (and therefore routes) from the corresponding symmetric network.

VI. HEURISTIC PROCEDURES FOR ARBITRARY NETWORKS

How does one go about choosing a "good" routing pattern to a given terminal node in an arbitrary network? We can begin by making the following observation.

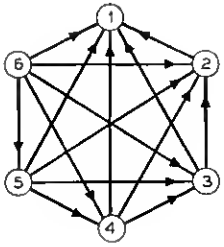

Fig. 9 — Six-node symmetric network.

Since a practical pattern defines routes to *one* node, the destination, it is reasonable to require "good" routing patterns to have only one terminating node type, the destination node. This means we may apply backward production as defined by rules 1 and 2; therefore, we cannot expect to generate *all* loop-free routing patterns (which require rule 1′ in place of rule 1). This is not a limitation unless we are dealing with multiple-homed stations. We will ignore this latter case, although the procedures we discuss can be generalized to deal with multiple homing.

Now, what is meant by a "good" routing pattern? One with the lowest average blocking from all nodes to the destination node? A pattern which minimizes blocking from a *selected* node to the destination? One with the smallest average route length? A pattern with the maximum total number of routes?*

To the author's knowledge no algorithm exists which will guarantee

* There is, of course, the larger problem of designing a network which realizes all of these and is, at the same time, rugged, economical, and so on, as described in the introduction. This is a complex problem; its very statement is difficult and has been the subject of intensive study, (See Refs. 3, 4 and 5.)

any of these criteria. However, it is possible to approach the last two criteria by using a heuristic procedure which will generate patterns with large numbers of short routes, and which also has the virtue of assigning orders of choice to the branches.

6.1 *Generating Patterns with Many Short Routes*

Consider the following method for applying rules 1 and 2 to an arbitrary network:

(*i*) Select the destination node as the first node and apply rule 1, labeling all the branches incoming. Label the originating ends of each of these branches the first choice out of the respective nodes.

(*ii*) Now, in the set of nodes to which the destination node connects (these will be called "predecessors" of the destination node), select any node and apply rule 2, labeling its free branches incoming. Label the originating ends of these branches first choice out of the respective nodes, if possible; or, if a first choice already exists (from step *i*) label the branch second choice.

(*iii*) Continue step *ii*, choosing nodes only from the predecessors of the destination node; each time, label the branches n plus first choice out of the node at the originating end, where n choices already exist. Continue until all the predecessors of the destination node have had rule 2 applied to them.

(*iv*) Consider the set of nodes which has outgoing branches to any node (or nodes) which are predecessors of the destination node.† These may be thought of as second level predecessors of the destination node. Apply rule 2 to these nodes until they have been exhausted (or until you are exhausted, whichever comes first), each time labeling branches the n plus first choice out of the node in which they originate.

(*v*) Identify the third level predecessors of the destination node, and so on. Continue the process until every branch in the graph has a direction and order of choice out of the node from which it originates.

Fig. 10 gives an example of the procedure, which is tedious to describe, but easy to perform.

At this point, we can observe that all the paths from the K^{th} level predecessors of the destination node have at least K links. We prove the following theorem:

† The predecessors of any node (or nodes) can be identified without reference to branch directions. In this procedure, a predecessor of node A is any node connected to node A by an (as yet) undirected branch. If we are seeking the predecessors of a group of nodes, branches between nodes in the group are ignored.

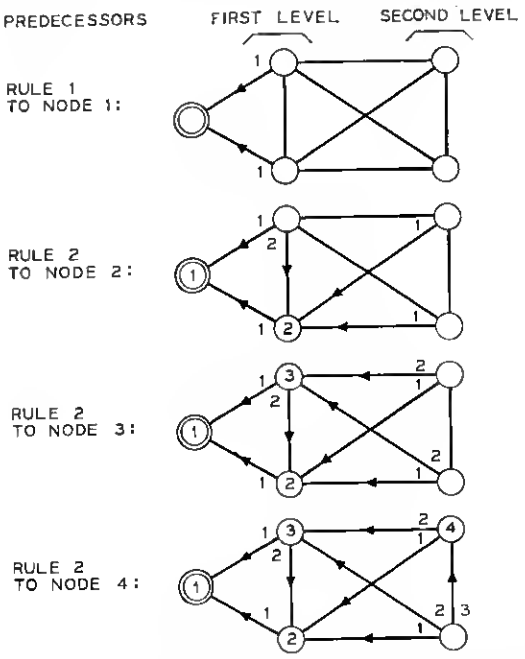

Fig. 10 — Deriving order of choice in the backward production process.

Theorem 4: The given procedure creates the maximum number of 1-link and 2-link routes.

Proof: Clearly, there is no way to create more 1-link paths to the destination node than to label all its branches incoming. Consider the first level predecessors of the destination node. Among their free branches, they may have branches to each other, and branches from second level predecessors.

Each time we label a branch between first level predecessors we create one 2-link path to the destination. This is true regardless of the direction given to the branch. Hence, the number of 2-link paths created this way is fixed, and is exactly the number of branches between first level predecessors. If we now discard the branches between first level predecessors and consider the reduced graph, it is clear that the way to get the maximum number of 2-link paths is to label every free branch, on every first level predecessor, incoming. But this is exactly the effect of the given procedure. Branches that do not connect first level predecessors remain free until rule 2 is applied to the node; then they are all labeled incoming. It follows that the total

number of 2-link paths created is fixed, and is equal to the number of free branches on all first level predecessors after the application of rule 1 to the destination node. The order in which rule 2 is applied to these nodes has no effect on the number of 2-link routes.

It might seem that this theorem can be extended to show that the procedure produces the maximum number of K-link routes ($K > 2$), subject to the fact that K-1 link, K-2 link, . . . , 2-link, and 1-link routes have been maximized. Unfortunately, one need go no higher than 3-link routes to find a counterexample as shown in Fig. 11.

The heuristic procedure can be improved by eliminating the arbitrary choosing of nodes in step *ii* and in later steps. That is, having identified the N^{th} level predecessors of the destination node, we apply rule 2 to these nodes in a particular order.

6.2 *Choosing N^{th} Level Predecessors*

We suggest this revised heuristic procedure for choosing among N^{th} level predecessors:

(*i*) Arrange the graph to show the various level predecessors in stages. An example is Fig. 12, where higher and higher level predecessors are encountered as we progress from left to right.

(*ii*) Direct all branches between stages toward the destination node. (See Fig. 12.)

(*iii*) Now consider the first level predecessors, nodes 2, 3, and 4.

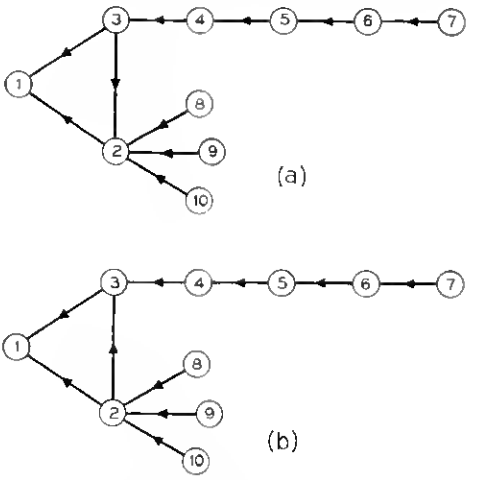

Fig. 11 — Counterexample. (a) Pattern generated by heuristic procedure; number of 3-link routes: 2. (b) Pattern with maximum number of 3-link routes; number of 3-link routes: 4.

For each of these nodes, we compile two figures, the number of routes provided from the node to the destination, and the number of nodes served by this node. Node B is said to be served by node A if there exists at least one directed path from B to A. Node 2, for example, serves nodes 5 and 9, while node 7 serves none. We take the difference of these two numbers (nodes served minus routes provided) and use the resulting number as a measure of the need for additional routes. For Fig. 12b these numbers may be tabulated as follows:

Node	No. Nodes Served	No. Routes Provided	Differenee
2	2	1	1
3	3	1	2
4	3	1	2

(*iv*) Now choose the lowest of the difference numbers and apply rule 2 to the corresponding node. In this example, node 2 is the choice and we label all its branches incoming. (Presumably, it needs the least number of additional routes.)

(*v*) Node 2 is now removed from consideration and we may restate the table for nodes 3 and 4, adding the routes picked up by the branches directed into node 2:

Node	No. Nodes Served	No. Routes Provided	Difference
3	3	2	1
4	3	2	1

In this case, we have equality and so choose node 3 arbitrarily. Node 4 is, then, the last node in the process and the result is shown in Fig. 12c.

(*vi*) We now move one stage to the right and consider second level predecessors:

Node	No. Nodes Served	No. Routes Provided	Differenee
5	1	1	0
6	2	2	0
7	0	4	−4
8	1	4	−3

This suggests that node 7 is least in need of additional routes and we may label all its branches incoming. Restating the table two more

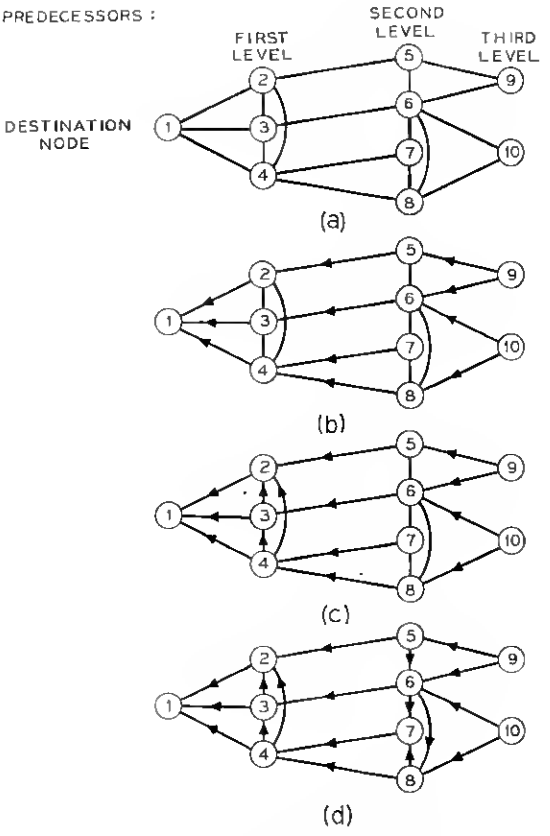

Fig. 12 — Example of heuristic procedure.

times, we obtain node 8 next and, finally, node 6. The result is shown in Fig. 12d.

Node	No. Nodes Served	No. Routes Provided	Difference
5	1	1	0
6	2	6	−4
8	1	8	−7
5	1	1	0
6	2	14	−12

To obtain an order of choice for the branches, we simply apply the heuristic procedure for generating patterns with large numbers of

short routes in node order 1, 2, 3, 4, 7, 8, 6, and 5. (See Section 6.1.) The result, identical to that in Fig. 12d, is shown in Fig. 13.

This method yields routing patterns in which the average path length is short and the total number of routes large. It is, however, not infallible, and counterexamples can be generated—networks in which the process leads to neither a minimum average path length nor a maximum total number of routes.

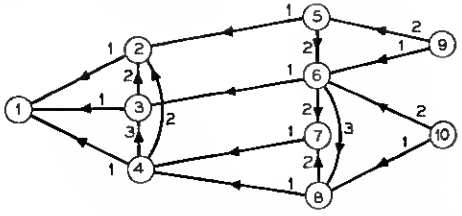

Fig. 13 — The complete routing pattern to node 1. Apply backward production in the order: 1, 2, 3, 4, 7, 8, 6, 5.

VII. SUMMARY AND CONCLUSIONS

This paper discusses methods for generating loop-free directed routing patterns and for detecting the presence of loops in arbitrary patterns. The heuristic procedures suggested seem to yield useful patterns for the size network that can be considered by hand; moreover, they are clearly programmable, thus allowing us to deal with large networks.

The procedures and theorems we present are not addressed to the problem of achieving optimum traffic handling abilities of communication networks. They are, however, a preliminary step to such examinations and, hopefully, present an orderly and useful way of looking at the process of routing as it is currently practiced. These theorems and procedures suggest ways of modifying present routing practices which may be fruitful to explore.

REFERENCES

1. Weber, J. H., unpublished work.
2. Hakimi, S. L., "On the Degrees of the Vertices of a Directed Graph," J. Franklin Inst., *279*, No. 4 (April 1965), pp. 290–308.
3. Wernander, M. A., "Systems Engineering for Communications Networks," talk at IEEE summer general meeting and nuclear radiation effects conference, Toronto, Ont., Canada, June 16–21, 1963.
4. Beneš, V. E., unpublished work.
5. Beneš, V. E., "Programming and Control Problems Arising from Optimal Routing in Telephone Networks," talk at the First International Conference on Programming and Control, USAF Academy, Colorado, April 15–16, 1965.